THE UNIVERSITY *of* EDINBURGH

Edinburgh Research Explorer

Distance-Based Regularisation of Deep Networks for Fine-Tuning

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
International Conference on Learning Representations (ICLR 2021)

OPEN ACCESS

DISTANCE-BASED REGULARISATION OF DEEP NETWORKS FOR FINE-TUNING

Henry Gouk & Timothy M. Hospedales
School of Informatics
University of Edinburgh
{henry.gouk,t.hospedales}@.ed.ac.uk

Massimiliano Pontil
Department of Computer Science
University College London
m.pontil@ucl.ac.uk

ABSTRACT

We investigate approaches to regularisation during fine-tuning of deep neural networks. First we provide a neural network generalisation bound based on Rademacher complexity that uses the distance the weights have moved from their initial values. This bound has no direct dependence on the number of weights and compares favourably to other bounds when applied to convolutional networks. Our bound is highly relevant for fine-tuning, because providing a network with a good initialisation based on transfer learning means that learning can modify the weights less, and hence achieve tighter generalisation. Inspired by this, we develop a simple yet effective fine-tuning algorithm that constrains the hypothesis class to a small sphere centred on the initial pre-trained weights, thus obtaining provably better generalisation performance than conventional transfer learning. Empirical evaluation shows that our algorithm works well, corroborating our theoretical results. It outperforms both state of the art fine-tuning competitors, and penalty-based alternatives that we show do not directly constrain the radius of the search space.

1 INTRODUCTION

The ImageNet Large Scale Visual Recognition Challenges have resulted in a number of neural network architectures that obtain high accuracy when trained on large datasets of labelled examples (He et al., 2016; Tan and Le, 2019; Russakovsky et al., 2015). Although these models have been shown to achieve excellent performance in these benchmarks, in many real-world scenarios such volumes of data are not available and one must resort to fine-tuning an existing model: taking the weights from a model trained on a large dataset, to initialise the weights for a model that will be trained on a small dataset. The assumption being that the weights from the pre-trained model provide a better initialisation than randomly generated weights. Approaches for fine-tuning are typically ad hoc, requiring one to experiment with many problem-dependent tricks, and often a process that will work for one problem will not work for another. Transforming fine-tuning from an art into a well principled procedure is therefore an attractive prospect. This paper investigates, from both a theoretical and empirical point of view, the impact of different regularisation strategies when fine-tuning a pre-trained network for a new task.

Existing fine-tuning regularisers focus on augmenting the cross entropy loss with terms that indirectly or directly penalise the distance the fine-tuned weights move from the pre-trained values. The intuition behind this seems sensible—the closer the fine-tuned weights are to the pre-trained weights, the less information is forgotten about the source dataset—but it is not obvious how this idea should be translated into an effective algorithm. One should expect that the choice of distance metric is quite important, but existing methods exclusively make use of Euclidean distance (Li et al., 2019; 2018) without any theoretical or empirical justification regarding why that metric was chosen. These methods achieve only a small improvement in performance over standard fine-tuning, and it is reasonable to expect that using a metric more suited to the weight space of neural networks would lead to greater performance. Moreover, while the use of penalty terms to regularise neural networks is well established, the impact of using penalties vs constraints as regularisers has not been well studied in the context of deep learning.

In order to study the generalisation error of fine-tuned models, we derive new bounds on the empirical Rademacher complexity of neural networks based on the distance the trained weights move from

their initial values. In contrast to existing theory (e.g., Neyshabur et al. (2018); Bartlett et al. (2017); Long and Sedghi (2019)), we do not resort to covering numbers or make use of distributions over models to make these arguments. By deriving two bounds utilising different distance metrics, but proved with the same techniques, we are able to conduct a controlled theoretical comparison of which metric one should use as the basis for a fine-tuning regularisation scheme. Our findings show that a metric based on the maximum absolute row sum (MARS) matrix norm is a more suitable measure of distance in the parameter space of convolutional neural networks than Euclidean distance. Additionally, we challenge the notion that using a penalty term to encourage the fine-tuned weights to lie near the pre-trained values is the best way to restrict the effective hypothesis class. We demonstrate that the equivalence of penalty methods and constraint methods in the case of linear models (Oneto et al., 2016) does not translate to the context of deep learning. As a result, using projected stochastic subgradient methods to constrain the distance the weights in each layer can move from the initial settings can lead to improved performance.

Several regularisation methods are proposed, with the aim of both corroborating the theoretical analysis with empirical evidence, and improving the performance of fine-tuned networks. One of these approaches is a penalty-based method that regularises the distance from initialisation according to the MARS-based distance metric. The other two techniques make use of efficient projection functions to enforce constraints on the Euclidean and MARS distance between the pre-trained and fine-tuned weights throughout training. The experimental results demonstrate that projected subgradient methods improve performance over using penalty terms, and that the widely used Euclidean metric is typically not the best choice of metric to measure distances in network parameter space.

2 RELATED WORK

The idea of developing an algorithm to restrict the distance of weights from some unbiased set of reference weights has been explored in various forms to improve the performance of fine-tuned networks. Li et al. (2018) presented the ℓ^2-SP regulariser, which consists of adding a term to the objective function that penalises the squared ℓ^2 distance of the trained weights from the initial weights. This is based on an idea initially made use of when performing domain adaptation, where it was applied to linear support vector machines (Yang et al., 2007). The subsequent work of Li et al. (2019) follows the intuition that the *features* produced by the fine-tuned network should not differ too much from the pre-trained features. They also use Euclidean distance, but to measure distance between feature vectors rather than weights. The idea is extended to incorporate an attention mechanism that weights the importance of each channel. The method is implemented by adding a penalty term to the standard objective function. In contrast to these approaches, we solve a constrained optimisation problem rather than adding a penalty, and we demonstrate that the MARS norm is more effective than the Euclidean norm when measuring distance in weight space.

Many recent meta-learning algorithms also make use of idea that keeping fine-tuned weights close to their initial values is desirable. However, these approaches typically focus on developing methods for learning the initial weights, rather than working with pre-specified initial weights. The model-agnostic meta-learning approach (Finn et al., 2017) does this by simulating few-shot learning tasks during the meta-learning phase in order to find a good set of initial weights for a neural network. Once the learned algorithm is deployed, it adapts to new few-shot learning tasks by fine-tuning the initial weights for a small number of iterations. Denevi et al. (2018) proposes a modified penalty term for ridge regression where, instead of penalising the distance of the parameters from zero, they are regularised towards a bias vector. This bias vector is learned during the course of solving least squares problems on a collection of related tasks. Denevi et al. (2019) extend this approach to a fully online setting and a more general family of linear models.

Previous work investigating the generalisation performance of neural network based on the distance the weights have travelled from their initial values has done so with the aim of explaining why existing methods for training models work well. Bartlett et al. (2017) present a bound derived via covering numbers that shows the generalisation performance of fully connected networks is controlled by the distance the trained weights are from the initial weights. Their bound makes use of a metric that scales with the number of units in the network, which means if the result is extended to a class of simple convolutional networks then the generalisation performance will scale with the resolution of the feature maps. A similar bound can also be proved through the use

of PAC-Bayesian analysis (Neyshabur et al., 2018). One can make use of different metrics and techniques for applying covering numbers to bounding generalisation that do not have the same implicit dependence on the number of units, but they instead depend directly on the number of weights in the network (Long and Sedghi, 2019). Neyshabur et al. (2019) investigate the performance of two layer neural networks with ReLU activation functions, demonstrating that as the size of the hidden layer increases, the Frobenius distance (i.e., the Frobenius norm of the difference between initial and trained weight matrices) shrinks. Inspired by this observation, they show that one can construct a bound on the Rademacher complexity of this class by using Euclidean distance between the initial weights and trained weights of each individual unit. Although they bound the Rademacher complexity directly, they still incur an explicit dependence on the size of the hidden layer, and their analysis is restricted to fully connected networks with only a single hidden layer.

In contrast to these previous studies, our focus is on designing an algorithm that will improve the performance of fine-tuned networks, rather than explaining the performance of the standard fine-tuning methods that are already widespread. Therefore, we do not aim to infer what properties of existing methods enable networks to generalise well—we instead derive such properties and then develop algorithms that enforce them. Moreover, we put a particular emphasis on choosing a metric that is suitable for contemporary convolutional networks, and thus will not scale with the size of the feature maps, while also being easy to implement efficiently.

3 DISTANCE-BASED GENERALISATION BOUNDS

Throughout this section we will analyse a loss class consisting of feed-forward neural networks, $f(\vec{x}) = (\phi_L \circ ... \circ \phi_1)(\vec{x})$, where each $\phi_j(\vec{x}) = \varphi(W_j \vec{x})$ is a layer with a 1-Lipschitz activation function, φ. Both the norm of the weight matrix of each layer and the distance of the fine-tuned weights from the pre-trained weights are bounded from above. Formally, we define

$$\mathcal{F}_* = \{(\vec{x}, y) \mapsto l(y, f(\vec{x})) : \|W_j\|_* \leq B_j^*, \|W_j^0\|_* \leq B_j^*, \|W_j - W_j^0\|_* \leq D_j^*\},$$

where j goes from 1 to L, l is a ρ-Lipschitz loss function with a range of $[0, 1]$, and W_j^0 is the pre-trained weight matrix for layer j. Each of the pre-trained weight matrices can be either random variables drawn from a distribution with support such that the constraints are always fulfilled, or they can be fixed (i.e., drawn from a Dirac delta distribution). The only other requirement is that they are independent of the training data used for fine-tuning. From an empirical perspective it is most useful to consider them non-random quantities. We have used $*$ as a placeholder for the norm used to measure the magnitude of weight matrices and the distance from the pre-trained weights. The primary focus is on using the MARS norm,

$$\|W\|_\infty = \max_j \sum_{i=1} |W_{j,i}|,$$

to prove a bound on the empirical Rademacher complexity (Bartlett and Mendelson, 2002) of $\mathcal{F}_\infty$. Our main theoretical result, presented below, is obtained by modifying the "peeling"-style arguments typically used to directly prove bounds on the Rademacher complexity of neural network hypothesis classes (for examples, see Neyshabur et al. (2015); Golowich et al. (2018)). Our modification of the argument allows us to rephrase the resulting theorem in terms of the distance the parameters can move from their initialisation during training.

Theorem 1. *For all $\delta \in (0, 1)$, the expected loss of all models in $\mathcal{F}_\infty$ is, with probability $1 - \delta$, bounded by*

$$\mathbb{E}_{(\vec{x}, y)}[l(f(\vec{x}), y)] \leq \frac{1}{m} \sum_{i=1}^m l(f(\vec{x}_i), y_i) + \frac{4\sqrt{\log(2d)}c\rho C_\infty \sum_{j=1}^L \frac{D_j^\infty}{B_j^\infty} \prod_{j=1}^L 2B_j^\infty}{\sqrt{m}} + 3\sqrt{\frac{\log(2/\delta)}{2m}},$$

where m is the number of training examples, c is the number of classes, $\vec{x}_i \in \mathbb{R}^d$, and $\|\vec{x}_i\|_\infty \leq C_\infty$.

Crucially, when l is chosen carefully (e.g., the ramp loss), the expectation in Theorem 1 is an upper bound for the expected classification error rate. The proof for Theorem 1 can be found in the supplemental material. The two main terms in this theorem are (i) the product of bounds on the layer norms; and (ii) the summation, which is a bound on the distance the fine-tuned weights can be from the pre-trained weights. The first of these is primarily dependent on the weights obtained via pre-training, whereas (ii) can be controlled during the fine-tuning process. Moreover, one would expect that if better initial weights are selected via pre-training, then the distance the final weights will

be from the initial values will be smaller, thus leading to better generalisation. This is the motivation behind the regularisers we develop in Section 4.

An analogous bound is also derived for the Frobenius norm (i.e., $\mathcal{F}_F$) to facilitate a theoretical comparison between the proposed regularisation method and ℓ^2-SP, an existing approach that relies on penalising Euclidean distance between the pre-trained and fine-tuned parameters. In order to avoid a direct dependence on the number of parameters (as accomplished in Theorem 1) it is necessary to restrict $\mathcal{F}_F$ to use only ReLU activation functions.

Theorem 2. *For all $\delta \in (0,1)$, the expected loss of all models in $\mathcal{F}_F$ is, with probability $1 - \delta$, bounded by*

$$\mathbb{E}_{(\vec{x},y)}[l(f(\vec{x}),y)] \leq \frac{1}{m}\sum_{i=1}^{m} l(f(\vec{x}_i),y_i) + \frac{2\sqrt{2}c\rho C_2 \sum_{j=1}^{L} \frac{D_j^F}{2B_j^F \prod_{i=1}^{j}\sqrt{n_i}} \prod_{j=1}^{L} 2B_j^F \sqrt{n_j}}{\sqrt{m}} + 3\sqrt{\frac{\log(2/\delta)}{2m}},$$

where m is the number of training examples, c is the number of classes, $\|\vec{x}_i\|_2 \leq C_2$, $\varphi(\cdot) = \mathrm{ReLU}(\cdot)$, and n_j is the number of columns in W_j.

We can make several observations by comparing our two bounds with each other and previously published bounds: (i) in contrast to Theorem 1, Theorem 2 incurs a significant dependence on the resolution of the intermediate feature maps due to the $\sqrt{n_j}$ factors; (ii) each term in the summations in our bounds can be at most one, whereas the corresponding terms in Bartlett et al. (2017) can be more than one due to the use of multiple types of norms; (iii) our bound from Theorem 1 does not have a direct dependence on the number of weights, in contrast to the bound provided by Long and Sedghi (2019). More detailed comparisons can be found in the supplemental material.

4 FINE-TUNING WITH DISTANCE REGULARISATION

The analysis presented in Section 3 suggests that the weights in fine-tuned models should be close to the pre-trained weights in order to achieve good generalisation performance. More specifically, the learning process should search only within a set of weights within a predefined distance from the pre-trained weights. We discuss two strategies for accomplishing this: using projected subgradient methods to enforce a hard constraint, and augmenting the standard cross entropy objective with a term that penalises the distance between the pre-trained and fine-tuned weights. The method based on projection functions is attractive because it guarantees that the constraints will be fulfilled even if one uses a heuristic optimisation method to train the network parameters. However, the penalty-based approaches are more common in the literature, and convenient from an implementation point of view due to the ubiquity of automatic differentiation. Nevertheless, in contrast to the projection-based methods, the techniques that use penalties have weaker assurances on whether a constraint is actually being enforced. We discuss the drawbacks of this non-equivalence further in the supplemental material.

4.1 OPTIMISING WITH PROJECTIONS

One way to enforce constraints on the weights of neural networks during training is to use a variant of the projected stochastic subgradient method. This is similar to typical stochastic subgradient methods used when training neural networks, but has the additional step of applying a projection operation after each weight update to ensure that the new weights lie inside the set of feasible parameters that satisfy the constraints. In the case of classic subgradient descent, in order to guarantee convergence towards a stationary point the projection function must perform a Euclidean projection,

$$\pi(\widehat{W}) = \arg\min_{W} \frac{1}{2}\|W - \widehat{W}\|_2^2 \tag{1}$$
$$\text{s.t. } g(W) \leq 0,$$

where $\widehat{W}$ are the newly updated parameters that may violate the constraint, W are the projected parameters, and $g(\cdot)$ is a convex function specifying the constraint. Although the Euclidean projection is required for the classic projected subgradient method, other optimisation algorithms may require the projection to be performed with respect to a different metric. Looking at different optimisers as instantiations of mirror descent with different Bregman divergences is one way to determine the type of projection that should be performed. Unfortunately some of the most common optimisers used in deep learning, such as Adam (Kingma and Ba, 2015), are not guaranteed to converge even when

there are no constraints on the parameters being optimised (Reddi et al., 2018). This makes extending it to perform constrained optimisation a purely heuristic endeavour, further compounded by the fact that it is also not clear which metric the projection should be performed with respect to. As such, the projections used in our approaches are performed with respect to the norm that is most convenient from an efficiency point of view.

Rather than attempting to constrain the distance between the all pre-trained and fine-tuned weights in the network using a single projection, constraints are applied on a layer-wise basis. This makes optimisation more manageable and also allows practitioners to favour fine-tuning certain parts of the network—e.g., if one is fine-tuning a network pre-trained on photos to perform a task on paintings where the same underlying classes are present, one might wish to allocate more fine-tuning capacity to earlier layers in the network. The types of constraints that we wish to enforce take the form

$$\|W_j - W_j^{(0)}\|_* \leq \gamma_j,$$

where γ_j is a hyperparameter that corresponds to the maximum allowable distance between the pre-trained weights and the fine-tuned weights for layer j. With a minor rearrangement, this yields a constraint specification in the form required for Equation 1,

$$g_j^*(W_j^{(0)}, W_j, \gamma_j) = \|W_j - W_j^{(0)}\|_* - \gamma_j,$$

where we have made the dependence on the pre-trained weights and the hyperparameter explicit. The resulting optimisation problem is

$$\min_{W_{1:L}} \sum_{i=1}^{m} l(y_i, (\phi_L \circ ... \circ \phi_1)(\vec{x}_i)) \tag{2}$$
$$\|W_j - W_j^{(0)}\|_* \leq \gamma_j \quad \forall j \in \{1 ... L\}.$$

The remainder of this section presents derivations for the projection functions, π_F and π_∞, corresponding to this constraint specification when it is instantiated with the Frobenius norm and the MARS norm, respectively. We provide pseudocode in the supplementary material that illustrates how these projections are integrated to the neural network fine-tuning procedure when using a variant of the stochastic subgradient method. We refer to the Frobenius norm instantiation as ℓ^2-PGM and the MARS norm version as MARS-PGM, where the PGM indicates the use of projection gradient methods.

4.1.1 CONSTRAINING FROBENIUS DISTANCE

When using the Frobenius distance, Equation 1 can be rewritten as

$$\pi_F(W^{(0)}, \widehat{W}, \gamma) = \arg\min_W \frac{1}{2}\|W - \widehat{W}\|_F^2$$
$$\text{s.t. } \|W - W^{(0)}\|_F - \gamma \leq 0.$$

To simplify the problem, we can instead work on a translated version of the same parameter space where $W^{(0)}$ is the origin. Setting $\widehat{T} = \widehat{W} - W^{(0)}$ and $T = W - W^{(0)}$, the problem becomes

$$\pi_2(\widehat{T}, \gamma) = \arg\min_T \frac{1}{2}\|T - \widehat{T}\|_F^2$$
$$\text{s.t.} \|T\|_F - \gamma \leq 0,$$

which is the Euclidean projection onto the ℓ^2 ball with radius γ, and has the known closed form solution

$$\pi_2(\widehat{T}, \gamma) = \frac{1}{\max\left(1, \frac{\|\widehat{T}\|_F}{\gamma}\right)} \widehat{T}.$$

Expanding the definition of $\widehat{T}$ and translating back into the correct parameter space yields the Frobenius distance projection function,

$$\pi_F(W^{(0)}, \widehat{W}, \gamma) = W^{(0)} + \frac{1}{\max\left(1, \frac{\|\widehat{W} - W^{(0)}\|_F}{\gamma}\right)}(\widehat{W} - W^{(0)}). \tag{3}$$

4.1.2 Constraining MARS Distance

The constraint on the MARS distance can be equivalently expressed as a collection of constraints on the ℓ^1 distance of each row in the weight matrix from the corresponding row in the pre-trained weight matrix. That is,

$$\|W - W^{(0)}\|_\infty \leq \gamma \iff \|\vec{w}_i - \vec{w}_i^{(0)}\|_1 \leq \gamma \quad \forall i,$$

where $\vec{w}_i$ is the ith row of W. One can then make use of the same translation trick used to derive the Frobenius distance projection function to change the ℓ^1 distance constraints to ℓ^1 norm constraints,

$$\pi_1(\widehat{\vec{t}_i}, \gamma) = \arg\min_{\vec{t}_i} \frac{1}{2}\|\vec{t}_i - \widehat{\vec{t}_i}\|_2^2 \tag{4}$$

$$\text{s.t. } \|\vec{t}_i\|_1 - \gamma \leq 0,$$

where $\vec{t} = \vec{w}_i - \vec{w}_i^{(0)}$. The problem in Equation 4 is a Euclidean projection onto the ℓ^1 ball with radius γ, for which there is no known closed form solution. There exist algorithms to find the ℓ^1 projection in time linearly proportional to the dimensionality of the vector (Duchi et al., 2008), but they are not amenable to implementation on graphics processing units due to the sequential nature of the computations involved. Instead, we apply a projection that minimises the ℓ^1 distance between the original point and its projection, subject to the projected point lying inside the ℓ^1 ball with radius γ,

$$\pi_1(\widehat{\vec{t}_i}, \gamma) = \frac{1}{\max(1, \frac{\|\widehat{\vec{t}_i}\|_1}{\gamma})} \widehat{\vec{t}_i}.$$

This projection, while not providing the closest feasible point measured in Euclidean distance, still provides a point that satisfies the constraints, but is trivial to implement efficiently. Finally, the projection function for the entire weight matrix is given by applying π_1 row-wise, and translating back into the correct parameter space,

$$\pi_\infty(W^{(0)}, \widehat{W}, \gamma) = \begin{bmatrix} \pi_1(\widehat{\vec{w}_1} - \vec{w}_1^{(0)}, \gamma) + \vec{w}_1^{(0)} \\ \vdots \\ \pi_1(\widehat{\vec{w}_n} - \vec{w}_n^{(0)}, \gamma) + \vec{w}_n^{(0)} \end{bmatrix},$$

where $\widehat{W}$ contains n rows.

4.2 Penalty Methods

One popular approach in the literature to encourage a model to not move too far from a set of initial weights is to augment the loss function with a penalty term. In our case, this would involve taking the standard objective for the problem at hand (e.g., cross entropy or the hinge loss), and adding penalty terms corresponding to each layer,

$$\min_{W_{1:L}} \sum_{i=1}^m l(y_i, f(\vec{x}_i)) + \sum_{j=1}^L \lambda_j \|W_j - W_j^{(0)}\|_*, \tag{5}$$

where λ_j are hyperparameters used to balance the regularisation terms with the main loss function. Due to the subdifferentiability of the two norms considered in this paper, instantiations of Equation 5 can be trained via automatic differentiation and a variant of the stochastic subgradient method. For the Frobenius norm, we actually penalise the squared Frobenius norm, which recovers the ℓ^2-SP approach of Li et al. (2018). We refer to the instantiation that penalises the MARS distance as MARS-SP.

5 Experiments

This section provides an empirical investigation into the predictive performance of the proposed methods relative to existing approaches for regularising fine-tuning, and also conducts experiments to demonstrate which properties of the novel algorithms are responsible for the change in performance. Two network architectures are used: ResNet-101 (He et al., 2016), which is representative of a typical large neural network, and EfficientNetB0 (Tan and Le, 2019), a leading architecture intended for use on mobile devices. Both networks are pre-trained on the 2012 ImageNet Large Scale Visual Recognition Challenge dataset (Russakovsky et al., 2015). The Adam optimiser is used for all experiments (Kingma and Ba, 2015). Information regarding the datasets and hyperparameter optimisation procedure can be found in the supplemental material.

Table 1: Results obtained with different regularisation approaches when fine-tuning ResNet-101 (top) and EfficientNetB0 (bottom) models pre-trained on the ILSVRC-2012 subset of ImageNet. We report the mean $\pm$ std. dev. of accuracy measured across five different random seeds.

Regularisation	Aircraft	Butterfly	Flowers	Pets	PubFig	DTD	Caltech	Avg. Rank
None	51.81±0.87	70.02±0.16	76.68±1.07	84.19±0.34	75.36±0.67	66.15±0.55	75.32±0.61	6.71
DELTA (Li et al., 2019)	60.38±1.26	77.91±0.24	86.57±0.27	88.11±0.52	82.23±2.48	69.38±0.68	78.88±0.27	4.14
LS (Müller et al., 2019)	60.86±0.65	76.52±0.23	86.49±0.38	**90.50±0.31**	84.38±0.47	68.08±0.81	**80.04±0.10**	3.14
ℓ^2-SP (Li et al., 2018)	60.16±1.33	76.56±0.19	83.11±0.27	86.23±0.41	83.79±3.69	69.87±0.24	79.94±0.17	4.00
MARS-SP	58.52±1.23	69.54±0.36	75.90±1.04	84.22±0.45	79.17±0.73	69.53±0.93	79.19±0.33	5.57
ℓ^2-PGM	69.61±0.15	77.97±0.13	86.91±0.66	90.47±0.33	84.14±0.81	**70.97±0.71**	78.45±0.13	2.57
MARS-PGM	**72.04±0.70**	**79.50±0.36**	**87.42±0.41**	89.23±1.36	**88.75±0.28**	70.23±0.53	79.11±0.19	**1.86**
Regularisation	Aircraft	Butterfly	Flowers	Pets	PubFig	DTD	Caltech	Avg. Rank
None	54.58±0.65	69.43±0.47	77.43±0.14	84.87±0.19	75.51±0.83	64.98±0.43	81.07±0.38	7.00
DELTA (Li et al., 2019)	70.61±0.18	79.61±0.21	84.60±0.23	89.27±0.22	88.62±0.54	71.37±0.35	81.53±0.38	3.57
LS (Müller et al., 2019)	56.14±0.27	71.52±0.14	83.42±0.70	84.95±0.55	77.05±0.31	65.53±0.26	83.12±0.17	5.43
ℓ^2-SP (Li et al., 2018)	69.26±0.26	78.88±0.27	86.61±0.48	89.79±0.28	84.04±0.98	71.12±0.52	82.91±0.39	3.71
MARS-SP	66.96±0.49	72.01±0.20	77.79±0.48	89.24±0.40	85.33±0.59	69.41±0.49	82.10±0.24	5.0
ℓ^2-PGM	70.87±0.33	81.81±0.22	86.95±0.17	89.33±0.19	88.45±0.36	71.63±0.59	84.30±0.09	2.29
MARS-PGM	**75.22±0.34**	**82.32±0.10**	**90.36±0.15**	**91.38±0.24**	**90.30±0.40**	**73.57±0.38**	**84.84±0.17**	**1.00**

5.1 PREDICTIVE PERFORMANCE

The first set of experiments are a performance comparison of the proposed methods and existing regularisation approaches for fine-tuning. The baselines considered are standard fine-tuning with no specialised regularisation, ℓ^2-SP (Li et al., 2018), DELTA (Li et al., 2019), and label smoothing (LS) as formalised in Müller et al. (2019). Once hyperparameters are obtained, each network architecture and regulariser combination is fine-tuned on both the training and validation folds of each dataset. The fine-tuning process is repeated five times with different random seeds to measure the robustness of each method to the composition of minibatches and initialisation of the final linear layer, which is trained from scratch. Test set accuracy is reported for the ResNet-101 and EfficientNetB0 architectures in Table 1.

Comparing the average ranks of the methods across different datasets (Demšar, 2006), the most salient trend is that the projection-based methods exhibit a significant increase in accuracy over their penalty counterparts that use the same distance metrics. This suggests that, in the case of fine-tuning, using a projection to enforce a constraint on the weights throughout the training process is a better regularisation strategy than adding a term to the objective function that penalises the deviations of weights from their pre-trained values. Additionally, looking further at the relative performance of the two projection-based regularisers, we can see that the MARS distance variant is more often a better choice than the Frobenius distance. This observation further supports the conclusions of the comparison of the bounds in Section 3.

5.2 DISTANCE FROM INITIALISATION

To further investigate the relationship between the penalty and projection strategies for regularisation, we analyse the distances between the pre-trained and fine-tuned weights at a per-layer basis. Figure 1 provides histograms indicating the distributions of per-layer MARS distances for the models without any regularisation, regularised with MARS-SP, trained with the MARS-PGM, and also DELTA. All networks were trained on the Pets dataset with the same hyperparameters used by the models examined in Section 5.1. We observe three trends from these plots. Firstly, both the penalty and projection methods reduce the MARS distance between the pre-trained and fine-tuned parameters relative to the unregularised model. Secondly, a significant number of the constraints enforced by the projection method are activated—i.e., many of the weight matrices lie on the boundary of the feasible set. In contrast, the penalty-based regulariser does not enforce similarly activated constraints. Finally, the results demonstrate that the DELTA method of Li et al. (2019) does not operate by implicitly regularising the same quantity, as its MARS distance histogram is longer tailed than the others.
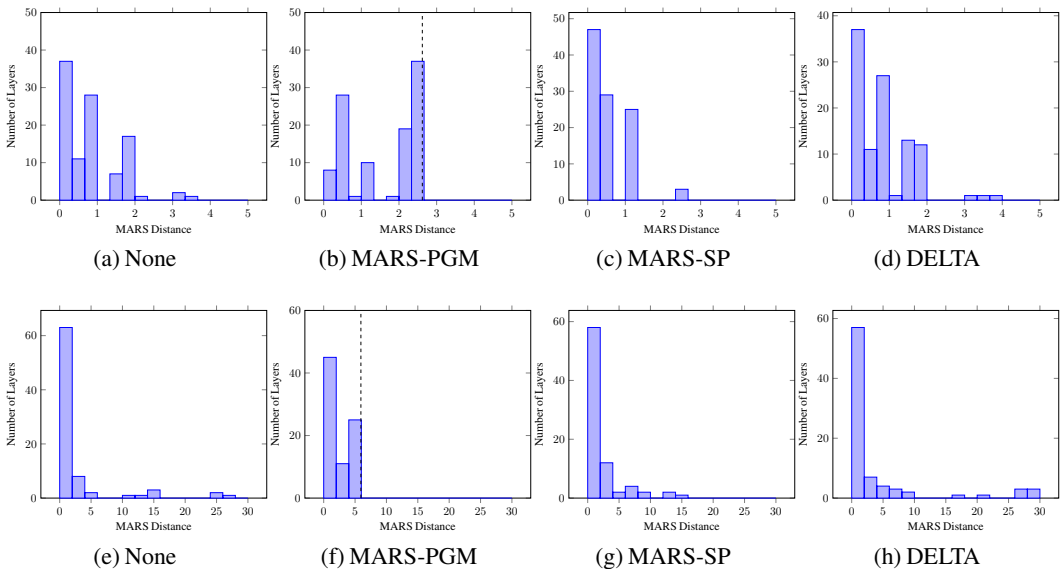
Figure 1: Histograms of MARS distances between pre-trained and pets dataset fine-tuned weights for the specified regularisation strategies. MARS-PGM successfully constrains weight distances to be less than γ_j, indicated by the dashed line. Top: ResNet101, Bottom: EfficientNetB0.
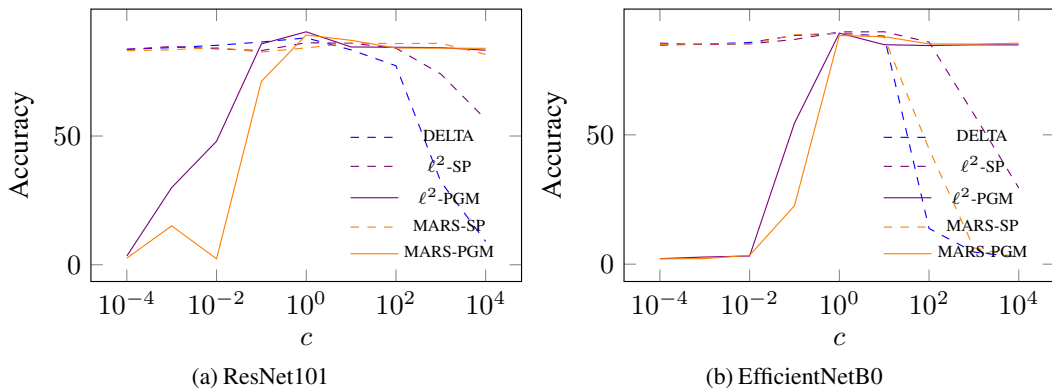

Figure 2: Sensitivity of the regularisation methods to the choice of hyperparameters. Measurements are taken on the pets dataset, and c is a factor applied to the hyperparameters found during tuning.

5.3 CAPACITY CONTROL

To demonstrate the ability of the distance-based regularisation methods to control model capacity, we sweep through a range of hyperparameter values and plot the corresponding predictive performance. Hyperparameters were generated according to $\lambda_j = c\hat{\lambda}_j$ and $\gamma_j = c\hat{\gamma}_j$, where c is varied, and $\hat{\lambda}_j, \hat{\gamma}_j$ are the values found during the hyperparameter optimisation process. Plots of c versus the resulting accuracy on the pets dataset are given in Figure 2 for both the ResNet101 and EfficientNetB0 architectures. We can see that the PGM methods behave as the theoretical analysis predicts: hyperparameter configurations that lead to very small distances between pre-trained and fine-tuned weights result in underfitting, relaxing the hyperparameters too much leads to overfitting, and using the optimised hyperparameters (i.e., when $c = 1$) achieves the best performance.

(a) ℓ^2-PGM   (b) MARS-PGM   (c) Unregularised   (d) Unregularised Accuracy
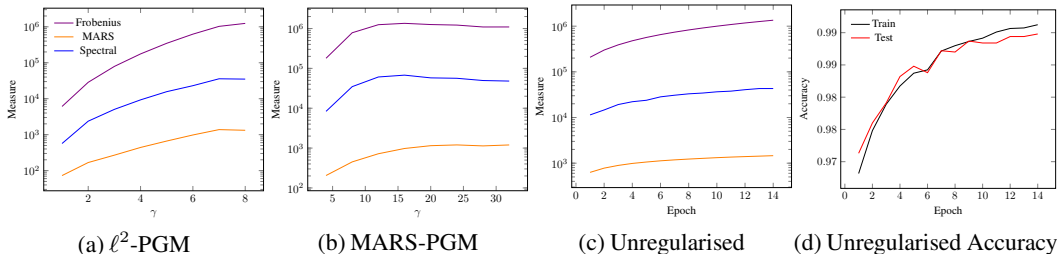
Figure 3: An empirical comparison of the tightness of our bounds on the Rademacher complexity, and that of Long and Sedghi (2019). The (a) and (b) plots demonstrate the empirical value of the bounds as a function of the regularisation parameters: the vertical axes correspond to the model complexities for each of the three measures considered, and the horizontal axes represents the regularisation strength for ℓ^2-PGM (a) and MARS-PGM (b). The plot in (c) shows how the model complexity measures change throughout the training of an unregularised model, and (d) shows the training and test set performance throughout training for the same unregularised model.

5.4 EMPIRICAL COMPARISON OF BOUNDS

We perform an empirical comparison of our two bounds, along with a bound based on the spectral norm (Long and Sedghi, 2019), to demonstrate the relative tightness. This is done by training neural networks on the MNIST dataset (LeCun et al., 1998). We use the architecture of Scherer et al. (2010), which consists of a single convolutional layer with 9×9 filters and 112 channels, followed by 5×5 max pooling, and finally a linear classifier layer. The network is trained for 15 epochs using the Adam optimiser (Kingma and Ba, 2015), as training any further does not result in any performance increase. Following similar work that has performed empirical evaluation of neural network bounds, we evaluate the bounds by computing the relevant norms and distances of the trained network weights. These quantities are then used in place of the upper bounds (i.e., B_i^* and D_i^*) used in the definition of the hypothesis classes considered by Theorems 1 and 2. In keeping with previous work that has performed this type of empirical comparison, we measure the distance from random initialisation, rather than fine-tuning from a pre-trained network (Bartlett et al., 2017; Neyshabur et al., 2019). Figure 3 shows how these three quantities vary for different choices of γ (the same hyperparameter is used for both layers) when using MARS-PGM and ℓ^2-PGM, and also how they differ through the process of training and unregularised model. We observe several trends: (i) like previous work in this area, all of the empirically evaluated Rademacher complexity bounds are still too loose to be useful for model selection and providing performance guarantees on the expected test set performance; (ii) the bounds based on the MARS norm are consistently tighter (by orders of magnitude) than the bounds based on the Frobenius and spectral norms; (iii) the empirical measurements of model complexity plateau even though the hyperparameters governing the worst-case capacity continue to increase—we suspect this is caused by implicit regularisation from early stopping. This implicit regularisation from early stopping also provides an explanation for why we observe only a small degradation in performance in Figure 2 when the hyperparameters are set to very large values.

6 CONCLUSION

This paper investigates different regularisation methods for fine-tuning deep learning networks. To facilitate this, we provide two new bounds on the generalisation performance of neural networks based on the distance of the final weights from their initial values. The discussion comparing these bounds suggests that the MARS distance is a more appropriate metric in the parameter space of convolutional networks than Frobenius distance. Additionally, several new algorithms are presented that enable an experimental comparison between different regularisation strategies. The empirical results corroborate our theoretical investigation, demonstrating that constraining MARS distance is more effective than constraining Euclidean distance. Crucially, we also show that, in line with our theoretical results, enforcing a hard constraint throughout the entire training process on the distances

the parameters can move is far more effective than the widely used strategy of adding a penalty term to the objective function. Implementations of the methods used in this paper are available online.[1]

ACKNOWLEDGMENTS

REFERENCES

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems 30*, pages 6240–6249, 2017.

James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D. Cox. Hyperopt: A Python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008, July 2015.

Tianshui Chen, Wenxi Wu, Yuefang Gao, Le Dong, Xiaonan Luo, and Liang Lin. Fine-grained representation learning and recognition by exploiting hierarchical semantic embedding. In *Proceedings of the 26th ACM International Conference on Multimedia*, pages 2023–2031, 2018.

M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

Giulia Denevi, Carlo Ciliberto, Dimitris Stamos, and Massimiliano Pontil. Learning to learn around a common mean. In *Advances in Neural Information Processing Systems 31*, 2018.

Giulia Denevi, Carlo Ciliberto, Riccardo Grazzi, and Massimiliano Pontil. Learning-to-learn stochastic gradient descent with biased regularization. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-Independent Sample Complexity of Neural Networks. In *Conference On Learning Theory*, pages 297–299, July 2018.

Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Michel Ledoux and Michel Talagrand. *Probability in Banach Spaces Isoperimetry and Processes*. Springer, 1991.

---

[1]https://github.com/henrygouk/mars-finetuning

Xingjian Li, Haoyi Xiong, Hanchao Wang, Yuxuan Rao, Liping Liu, and Jun Huan. DELTA: DEep Learning Transfer using Feature Map with Attention for Convolutional Networks. In *International Conference on Learning Representations*, 2019.

Xuhong Li, Yves Grandvalet, and Franck Davoine. Explicit inductive bias for transfer learning with convolutional networks. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Philip M Long and Hanie Sedghi. Size-free generalization bounds for convolutional neural networks. *arXiv preprint arXiv:1905.12600*, 2019.

S. Maji, J. Kannala, E. Rahtu, M. Blaschko, A. Vedaldi, Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013.

Andreas Maurer. A vector-contraction inequality for rademacher complexities. In *International Conference on Algorithmic Learning Theory*, pages 3–17. Springer, 2016.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems 32*, 2019.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, 2015.

Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A PAC-Bayesian Approach to Spectrally-Normalized Margin Bounds for Neural Networks. In *International Conference on Learning Representations*, February 2018.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations*, 2019.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the 6th Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008.

Luca Oneto, Sandro Ridella, and Davide Anguita. Tikhonov, ivanov and morozov regularization for support vector machine learning. *Machine Learning*, 103(1):103–136, 2016.

Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

Nicolas Pinto, Zak Stone, Todd Zickler, and David Cox. Scaling up biologically-inspired computer vision: A case study in unconstrained face recognition on facebook. In *IEEE CVPR Workshops*, pages 35–42. IEEE, 2011.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of Adam and beyond. In *International Conference on Learning Representations*, 2018.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3): 211–252, December 2015.

Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *International Conference on Artificial Neural Networks*, pages 92–101. Springer, 2010.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Jun Yang, Rong Yan, and Alexander G Hauptmann. Adapting svm classifiers to data with shifted distributions. In *Proceedings of the 7th IEEE International Conference on Data Mining Workshops*, pages 69–76. IEEE, 2007.

A PROOF OF THEOREM 1

We begin with the definition of empirical Rademacher complexity.

Definition 1. *The empirical Rademacher complexity of a class, $\mathcal{C}$, is defined as*

$$\hat{\mathcal{R}}(\mathcal{C}) = \frac{1}{m}\mathbb{E}_{\vec{\sigma}}\left[\sup_{h\in\mathcal{C}}\sum_{i=1}^{m}\vec{\sigma}_i h(\vec{x}_i)\right],$$

where $\vec{\sigma}$ is a vector of independent Rademacher distributed random variables and m is the number of training examples.

Using a standard result from statistical learning theory, one can make use of the empirical Rademacher complexity of $\mathcal{F}_\infty$ to bound the generalisation error of models contained within the class.

Theorem 3 ((Shalev-Shwartz and Ben-David, 2014, p. 378)). *For a class, $\mathcal{C}$, containing functions mapping from some set to the interval $[a, b]$, for all $\delta \in (0, 1)$ the following holds with probability at least $1 - \delta$:*

$$\mathbb{E}_{\vec{x},y}\Big[h(y,\vec{x})\Big] \leq \frac{1}{m}\sum_{i=1}^{m}h(y_i,\vec{x}_i) + 2\hat{\mathcal{R}}(\mathcal{C}) + 3(b-a)\sqrt{\frac{\log(2/\delta)}{2m}},$$

where $(\vec{x}_i, y_i)$ are independent and identically distributed.

We will make use of the following lemma, which uses the "peeling" method of Neyshabur et al. (2015) to bound the Rademacher complexity of a network based on the product of weight matrix norms.

Lemma 1. *For $\mathcal{F}_\infty$ as defined in Section 3, the following inequality holds*

$$\mathbb{E}_{\sigma}\left[\sup_{W_{1:k}}\left\|\sum_{i=1}^{m}\sigma_i f_k(\vec{x}_i)\right\|_\infty\right] \leq \sqrt{2m\log(2d)}C_\infty\prod_{j=1}^{k}2B_j^\infty$$

where $f_k = \phi_k \circ ... \circ \phi_1$, $\vec{x}_i \in \mathbb{R}^d$, and $\|\vec{x}_i\|_\infty \leq C_\infty$.

Proof.

$$\mathbb{E}_{\sigma}\left[\sup_{W_{1:k}}\left\|\sum_{i=1}^{m}\sigma_i\varphi(W_k f_{k-1}(\vec{x}_i))\right\|_\infty\right]$$

$$= \mathbb{E}_{\sigma}\left[\sup_{W_{1:k-1},\vec{w}_k}\left|\sum_{i=1}^{m}\sigma_i\varphi(\vec{w}_k^T f_{k-1}(\vec{x}_i))\right|\right]$$

$$\leq 2\mathbb{E}_{\sigma}\left[\sup_{W_{1:k-1},\vec{w}_k}\left|\sum_{i=1}^{m}\sigma_i\vec{w}_k^T f_{k-1}(\vec{x}_i)\right|\right]$$

$$\leq 2\mathbb{E}_{\sigma}\left[\sup_{W_{1:k-1},\vec{w}_k}\|\vec{w}_k\|_1\left\|\sum_{i=1}^{m}\sigma_i f_{k-1}(\vec{x}_i)\right\|_\infty\right]$$

$$= 2\mathbb{E}_{\sigma}\left[\sup_{W_{1:k}}\|W_k\|_\infty\left\|\sum_{i=1}^{m}\sigma_i f_{k-1}(\vec{x}_i)\right\|_\infty\right]$$

$$= 2B_k^\infty\mathbb{E}_{\sigma}\left[\sup_{W_{1:k-1}}\left\|\sum_{i=1}^{m}\sigma_i f_{k-1}(\vec{x}_i)\right\|_\infty\right],$$

where the first inequality is due to the Lipschitz composition property of Rademacher complexities (Ledoux and Talagrand, 1991) and the second follows from Hölder's inequality. Iterating this process for all layers of the network results in

$$\mathbb{E}_{\sigma}\left[\sup_{W_{1:k}}\left\|\sum_{i=1}^{m}\sigma_i f_k(\vec{x}_i)\right\|_\infty\right] \leq \left(\prod_{j=1}^{k}2B_j^\infty\right)\mathbb{E}_{\sigma}\left\|\sum_{i=1}^{m}\sigma_i\vec{x}_i\right\|_\infty.$$

The expectation can be bounded via Massart's inequality (Shalev-Shwartz and Ben-David, 2014, p. 383) to conclude the proof. □

The following lemma demonstrating how to "split" the Rademacher complexity bound into terms associated with the fine-tuned and pre-trained weights will also be useful.

Lemma 2. *For $\mathcal{F}_\infty$ as given in Section 3, the following inequality holds*

$$\mathbb{E}_\sigma\left[\sup_{W_{1:k}} \vec{v}^T \sum_{i=1}^m \sigma_i f_k(\vec{x}_i)\right] \leq \sqrt{2m\log(2d)} C_\infty \|\vec{v}\|_1 D_k^\infty \prod_{j=1}^{k-1} 2B_j^\infty$$

$$+ \|\vec{v}\|_1 \max_j \mathbb{E}_\sigma\left[\sup_{W_{1:k-1}} \vec{w}_{k,j}^{0T} \sum_{i=1}^m \sigma_i f_{k-1}(\vec{x}_i)\right],$$

where f_k, $\vec{x}_i$, and C_∞ are as defined in Lemma 1, and $\vec{v}$ is a fixed vector in $\mathbb{R}^n$.

Proof.

$$\mathbb{E}_\sigma\left[\sup_{W_{1:k}} \vec{v}^T \sum_{i=1}^m \sigma_i f_k(\vec{x}_i)\right] = \mathbb{E}_\sigma\left[\sup_{W_{1:k}} \vec{v}^T \sum_{i=1}^m \sigma_i \varphi(W_k f_{k-1}(\vec{x}_i))\right]$$

$$\leq \sum_{j=1}^n \mathbb{E}_\sigma\left[\sup_{W_{1:k}} v_j \sum_{i=1}^m \sigma_i \varphi(\vec{w}_{k,j}^T f_{k-1}(\vec{x}_i))\right]$$

$$\leq \sum_{j=1}^n |v_j| \mathbb{E}_\sigma\left[\sup_{W_{1:k}} \sum_{i=1}^m \sigma_i \vec{w}_{k,j}^T f_{k-1}(\vec{x}_i)\right],$$

where the first inequality comes from moving the summation outside the supremum, and the second from multiplying the hypothesis class by a constant and applying the Lipschitz composition property of Rademacher complexities. This can be further bounded from above by

$$\sum_{j=1}^n |v_j| \mathbb{E}_\sigma\left[\sup_{W_{1:k}} \sum_{i=1}^m \sigma_i \vec{w}_{k,j}^T f_{k-1}(\vec{x}_i)\right]$$

$$= \sum_{j=1}^n |v_j| \mathbb{E}_\sigma\left[\sup_{W_{1:k}} \vec{w}_{k,j}^T \sum_{i=1}^m \sigma_i f_{k-1}(\vec{x}_i)\right]$$

$$= \sum_{j=1}^n |v_j| \mathbb{E}_\sigma\left[\sup_{W_{1:k}} (\vec{w}_{k,j} - \vec{w}_{k,j}^0 + \vec{w}_{k,j}^0)^T \sum_{i=1}^m \sigma_i f_{k-1}(\vec{x}_i)\right]$$

$$\leq \sum_{j=1}^n |v_j| \mathbb{E}_\sigma\left[\sup_{W_{1:k}} (\vec{w}_{k,j} - \vec{w}_{k,j}^0)^T \sum_{i=1}^m \sigma_i \vec{f}_{k-1}(\vec{x}_i)\right] + \sum_{j=1}^n |v_j| \mathbb{E}_\sigma\left[\sup_{W_{1:k-1}} \vec{w}_{k,j}^{0T} \sum_{i=1}^m \sigma_i f_{k-1}(\vec{x}_i)\right]$$

$$\leq \|\vec{v}\|_1 \max_j \mathbb{E}_\sigma\left[\sup_{W_{1:k}} (\vec{w}_{k,j} - \vec{w}_{k,j}^0)^T \sum_{i=1}^m \sigma_i \vec{f}_{k-1}(\vec{x}_i)\right] + \|\vec{v}\|_1 \max_j \mathbb{E}_\sigma\left[\sup_{W_{1:k-1}} \vec{w}_{k,j}^{0T} \sum_{i=1}^m \sigma_i f_{k-1}(\vec{x}_i)\right]$$

where the first inequality comes from moving a summation outside the supremum, and the second from Hölder's inequality. Because Rademacher complexities are always non-negative, we can omit the absolute value in the definition of the vector ∞-norm that one would expect to be enclosing each of the expectations. The first term can be bounded from above by moving the $\max_j$ inside the expectation and then applying Hölder's inequality and the definition of the MARS norm,

$$\|\vec{v}\|_1 \max_j \mathbb{E}_\sigma\left[\sup_{W_{1:k}} (\vec{w}_{k,j} - \vec{w}_{k,j}^0)^T \sum_{i=1}^m \sigma_i \vec{f}_{k-1}(\vec{x}_i)\right] \leq \|\vec{v}\|_1 D_k^\infty \mathbb{E}_\sigma\left[\sup_{W_{1:k-1}} \left\|\sum_{i=1}^m \sigma_i \vec{f}_{k-1}(\vec{x}_i)\right\|_\infty\right].$$

Applying Lemma 1 to the resulting expectation completes the proof. $\square$

We now prove Theorem 1.

Proof. The main result of Maurer (2016) tells us that

$$\hat{\mathcal{R}}(\mathcal{F}_\infty) \leq \frac{\sqrt{2}\rho}{m} \mathbb{E}_\sigma\left[\sup_{W_{1:L}} \sum_{j=1}^c \sum_{i=1}^m \sigma_{i,j}(\phi_L \circ ... \circ \phi_1)_j(\vec{x}_i)\right],$$

where $(\phi_L \circ ... \circ \phi_1)_j(\vec{x}_i)$ is the j-th component of the output of $(\phi_L \circ ... \circ \phi_1)(\vec{x}_i)$. Denoting $f_k = \phi_k \circ ... \circ \phi_1$ and the j-th row of W_L as $\vec{w}_{L,j}^T$, the summation over classes can be brought outside

13

the Rademacher complexity and ϕ_L expanded as

$$\hat{\mathcal{R}}(\mathcal{F}_\infty) \leq \sum_{j=1}^{c} \frac{\sqrt{2}\rho}{m} \mathbb{E}_\sigma\left[\sup_{W_{1:L}} \sum_{i=1}^{m} \sigma_i \varphi_j(\vec{w}_{L,j}^T f_{L-1}(\vec{x}_i))\right]$$

to which the Lipschitz contraction inequality can be applied to obtain

$$\sum_{j=1}^{c} \frac{\sqrt{2}\rho}{m} \mathbb{E}_\sigma\left[\sup_{W_{1:L}} \sum_{i=1}^{m} \sigma_i \vec{w}_{L,j}^T f_{L-1}(\vec{x}_i)\right]$$

$$= \sum_{j=1}^{c} \frac{\sqrt{2}\rho}{m} \mathbb{E}_\sigma\left[\sup_{W_{1:L}} (\vec{w}_{L,j} - \vec{w}_{L,j}^0 + \vec{w}_{L,j}^0)^T \sum_{i=1}^{m} \sigma_i f_{L-1}(\vec{x}_i)\right].$$

Hölder's inequality and the definition of the MARS norm can be used to further bound this as

$$\sum_{j=1}^{c} \frac{\sqrt{2}\rho}{m} \left(\mathbb{E}_\sigma\left[\sup_{W_{1:L}} \|\vec{w}_{L,j} - \vec{w}_{L,j}^0\|_1 \left\|\sum_{i=1}^{m} \sigma_i f_{L-1}(\vec{x}_i)\right\|_\infty\right] + \mathbb{E}_\sigma\left[\sup_{W_{1:L-1}} \vec{w}_{L,j}^{0T} \sum_{i=1}^{m} \sigma_i f_{L-1}(\vec{x}_i)\right] \right)$$

$$\leq \sum_{j=1}^{c} \frac{\sqrt{2}\rho}{m} \left(\mathbb{E}_\sigma\left[\sup_{W_{1:L}} D_L^\infty \left\|\sum_{i=1}^{m} \sigma_i f_{L-1}(\vec{x}_i)\right\|_\infty\right] + \mathbb{E}_\sigma\left[\sup_{W_{1:L-1}} \vec{w}_{L,j}^{0T} \sum_{i=1}^{m} \sigma_i f_{L-1}(\vec{x}_i)\right] \right),$$

of which the first expectation can be dealt with via Lemma 1, yielding

$$\hat{\mathcal{R}}(\mathcal{F}_\infty) \leq \frac{2\sqrt{\log(2d)}c\rho C_\infty \frac{D_L^\infty}{2B_L^\infty} \prod_{i=1}^{L} 2B_i^\infty}{\sqrt{m}} + \sum_{j=1}^{c} \frac{\sqrt{2}\rho}{m} \mathbb{E}_\sigma\left[\sup_{W_{1:L-1}} \vec{w}_{L,j}^{0T} \sum_{i=1}^{m} \sigma_i f_{L-1}(\vec{x}_i)\right].$$

Lemma 2 can be applied to the expectation, yielding

$$\mathbb{E}_\sigma\left[\sup_{W_{1:L}} \vec{w}_{L,j}^{0T} \sum_{i=1}^{m} \sigma_i f_{L-1}(\vec{x}_i)\right]$$

$$\leq \sqrt{2m\log(2d)}C_\infty\|\vec{w}_{L,j}^0\|_1 D_{L-1}^\infty \prod_{i=1}^{L-2} 2B_i^\infty + \|\vec{w}_{L,j}^0\|_1 \max_k \mathbb{E}_\sigma\left[\sup_{W_{1:L-2}} \vec{w}_{L-1,k}^{0T} \sum_{i=1}^{m} \sigma_i f_{L-2}(\vec{x}_i)\right]$$

$$\leq \sqrt{2m\log(2d)}C_\infty \frac{D_{L-1}^\infty}{2B_{L-1}^\infty} \prod_{i=1}^{L} 2B_i^\infty + B_L^\infty \max_k \mathbb{E}_\sigma\left[\sup_{W_{1:L-2}} \vec{w}_{L-1,k}^{0T} \sum_{i=1}^{m} \sigma_i f_{L-2}(\vec{x}_i)\right].$$

Noting that $\max_k \|\vec{w}_{l,k}^0\|_1 \leq B_l^\infty$, Lemma 2 can be recursively applied to the remaining expectation until the following bound is obtained

$$\hat{\mathcal{R}}(\mathcal{F}_\infty) \leq \frac{2\sqrt{\log(2d)}c\rho C_\infty \sum_{j=1}^{L} \frac{D_j^\infty}{B_j^\infty} \prod_{i=1}^{L} 2B_i^\infty}{\sqrt{m}} + \sum_{j=1}^{c} \frac{\sqrt{2}\rho \prod_{i=2}^{L} B_i^\infty}{m} \max_k \mathbb{E}_\sigma\left[\vec{w}_{1,k}^{0T} \sum_{i=1}^{m} \sigma_i \vec{x}_i\right].$$

The remaining expectation evaluates to zero irrespective of which k is selected by the max, hence nullifying the whole term, and an application of Theorem 3 completes the proof. □

B PROOF OF THEOREM 2

The proof of Theorem 2 follows the same structure as the proof for Theorem 1. For conciseness, some details are omitted due to their similarity with the previous proof. We first provide a lemma demonstrating the peeling argument for Frobenius norm-bounded networks.

Lemma 3. *For $\mathcal{F}_F$ as defined in Section 3, the following inequality holds*

$$\mathbb{E}_\sigma\left[\sup_{W_{1:k}} \left\|\sum_{i=1}^{m} \sigma_i f_k(\vec{x}_i)\right\|_2\right] \leq \sqrt{m}C_2 \prod_{j=1}^{k} 2B_j^F$$

where $f = \phi_k \circ ... \circ \phi_1$, φ is the ReLU activation function, and $\|\vec{x}_i\|_2 \leq C_2$.

Proof. Iteratively applying Lemma 1 from Golowich et al. (2018) results in

$$\mathbb{E}_\sigma\left[\sup_{W_{1:k}} \left\|\sum_{i=1}^{m} \sigma_i f_k(\vec{x}_i)\right\|_2\right] \leq \mathbb{E}_\sigma\left[\left\|\sum_{i=1}^{m} \sigma_i \vec{x}_i\right\|_2\right] \prod_{j=1}^{k} 2B_j^F,$$

and the expectation can be further bounded by

$$
\begin{aligned}
\mathbb{E}_{\sigma}\left[\left\|\sum_{i=1}^{m}\sigma_i\vec{x}_i\right\|_2\right] &= \mathbb{E}_{\sigma}\left[\sqrt{\left\|\sum_{i=1}^{m}\sigma_i\vec{x}_i\right\|_2^2}\right] \\
&\le \sqrt{\mathbb{E}_{\sigma}\left[\left\|\sum_{i=1}^{m}\sigma_i\vec{x}_i\right\|_2^2\right]} \\
&= \sqrt{\left\|\sum_{i=1}^{m}\vec{x}_i\right\|_2^2} \\
&= \sqrt{m}C_2.
\end{aligned}
$$

$\square$

Next, a lemma corresponding to the splitting method is given.

Lemma 4. *For $\mathcal{F}_F$ as given in Section 3, the following inequality holds*

$$
\begin{aligned}
\mathbb{E}_{\sigma}\left[\sup_{W_{1:k}}\vec{v}^T\sum_{i=1}^{m}\sigma_i f_k(\vec{x}_i)\right] &\le \sqrt{mn}C_2\|\vec{v}\|_2 D_k^F\prod_{j=1}^{k-1}2B_j^F \\
&\quad + \sqrt{n}\|\vec{v}\|_2\max_j\mathbb{E}_{\sigma}\left[\sup_{W_{1:k-1}}\vec{w}_{k,j}^{0T}\sum_{i=1}^{m}\sigma_i f_{k-1}(\vec{x}_i)\right],
\end{aligned}
$$

where f_k, $\vec{x}_i$, and C_2 are as defined in Lemma 3, and $\vec{v}$ is a fixed vector in $\mathbb{R}^n$.

Proof. From the proof for Lemma 2, we have

$$
\begin{aligned}
\mathbb{E}_{\sigma}\left[\sup_{W_{1:k}}\vec{v}^T\sum_{i=1}^{m}\sigma_i f_k(\vec{x}_i)\right] &\le \sum_{j=1}^{n}|v_j|\mathbb{E}_{\sigma}\left[\sup_{W_{1:k}}(\vec{w}_{k,j}-\vec{w}_{k,j}^0)^T\sum_{i=1}^{m}\sigma_i\vec{f}_{k-1}(\vec{x}_i)\right] \\
&\quad + \sum_{j=1}^{n}|v_j|\mathbb{E}_{\sigma}\left[\sup_{W_{1:k-1}}\vec{w}_{k,j}^{0T}\sum_{i=1}^{m}\sigma_i f_{k-1}(\vec{x}_i)\right],
\end{aligned}
$$

of which the first term be further bounded

$$
\begin{aligned}
\sum_{j=1}^{n}&|v_j|\mathbb{E}_{\sigma}\left[\sup_{W_{1:k}}(\vec{w}_{k,j}-\vec{w}_{k,j}^0)^T\sum_{i=1}^{m}\sigma_i\vec{f}_{k-1}(\vec{x}_i)\right] \\
&\le \|\vec{v}\|_2\sqrt{\sum_{j=1}^{n}\mathbb{E}_{\sigma}\left[\sup_{W_{1:k}}(\vec{w}_{k,j}-\vec{w}_{k,j}^0)^T\sum_{i=1}^{m}\sigma_i\vec{f}_{k-1}(\vec{x}_i)\right]^2} \\
&\le \|\vec{v}\|_2\sqrt{\sum_{j=1}^{n}\mathbb{E}_{\sigma}\left[\sup_{W_{1:k}}D_k^F\left\|\sum_{i=1}^{m}\sigma_i\vec{f}_{k-1}(\vec{x}_i)\right\|_2\right]^2} \\
&= \sqrt{n}\|\vec{v}\|_2 D_k^F\mathbb{E}_{\sigma}\left[\sup_{W_{1:k}}\left\|\sum_{i=1}^{m}\sigma_i\vec{f}_{k-1}(\vec{x}_i)\right\|_2\right] \\
&\le \sqrt{mn}C_2\|\vec{v}\|_2 D_k^F\prod_{j=1}^{k-1}2B_j^F,
\end{aligned}
$$

15

where the first two inequalities are due to Cauchy-Schwarz, and the third is via Lemma 3. Moving on to the second term, we begin by applying Cauchy-Schwarz,

$$
\sum_{j=1}^{n} |v_j| \mathbb{E}_{\sigma} \left[\sup_{W_{1:k-1}} \vec{w}_{k,j}^{0T} \sum_{i=1}^{m} \sigma_i f_{k-1}(\vec{x}_i) \right] \leq \|\vec{v}\|_2 \sqrt{ \sum_{j=1}^{n} \mathbb{E}_{\sigma} \left[\sup_{W_{1:k-1}} \vec{w}_{k,j}^{0T} \sum_{i=1}^{m} \sigma_i \vec{f}_{k-1}(\vec{x}_i) \right]^2 }
$$

$$
\leq \|\vec{v}\|_2 \sqrt{ n \max_{j} \mathbb{E}_{\sigma} \left[\sup_{W_{1:k-1}} \vec{w}_{k,j}^{0T} \sum_{i=1}^{m} \sigma_i \vec{f}_{k-1}(\vec{x}_i) \right]^2 }
$$

$$
= \sqrt{n} \|\vec{v}\|_2 \max_{j} \mathbb{E}_{\sigma} \left[\sup_{W_{1:k-1}} \vec{w}_{k,j}^{0T} \sum_{i=1}^{m} \sigma_i \vec{f}_{k-1}(\vec{x}_i) \right],
$$

which concludes the proof. $\qquad\square$

Finally, we prove Theorem 2.

Proof. The proof for this theorem follows the same process as the proof for Theorem 1. The main result of Maurer (2016) is used to obtain

$$
\hat{\mathcal{R}}(\mathcal{F}_F) \leq \frac{\sqrt{2}\rho}{m} \mathbb{E}_{\sigma} \left[\sup_{W_{1:L}} \sum_{j=1}^{c} \sum_{i=1}^{m} \sigma_{i,j} (\phi_L \circ ... \circ \phi_1)_j(\vec{x}_i) \right],
$$

which, by bringing the summation over c outside the expectation and using the Lipschitz contraction inequality, can be bounded from above by

$$
\sum_{j=1}^{c} \frac{\sqrt{2}\rho}{m} \mathbb{E}_{\sigma} \left[\sup_{W_{1:L}} (\vec{w}_{L,j} - \vec{w}_{L,j}^0 + w_{L,j}^0)^T \sum_{i=1}^{m} \sigma_i f_{L-1}(\vec{x}_i) \right].
$$

This can be expanded into two expectations, as in the proof for Theorem 1, and Lemma 3 applied to the first term, yielding

$$
\hat{\mathcal{R}}(\mathcal{F}_F) \leq \frac{\sqrt{2}c\rho C_2 \frac{D_L^F}{B_L^F} \prod_{i=1}^{L} 2B_i^F}{\sqrt{m}} + \sum_{j=1}^{c} \frac{\sqrt{2}\rho}{m} \mathbb{E}_{\sigma} \left[\sup_{W_{1:L-1}} \vec{w}_{L,j}^{0T} \sum_{i=1}^{m} \sigma_i f_{L-1,k}(\vec{x}_i) \right].
$$

Applying Lemma 4 to the expectation results in

$$
\mathbb{E}_{\sigma} \left[\sup_{W_{1:L}} \vec{w}_{L,j}^{0T} \sum_{i=1}^{m} \sigma_i f_{L-1}(\vec{x}_i) \right]
$$

$$
\leq \sqrt{mn_L} C_2 \|\vec{w}_{L,j}^0\| D_{L-1}^F \prod_{i=1}^{L-2} 2B_i^F + \sqrt{n_L} \|\vec{w}_{L,j}^0\|_2 \max_{k} \mathbb{E}_{\sigma} \left[\sup_{W_{1:L-1}} \vec{w}_{L-1,k}^{0T} \sum_{i=1}^{m} \sigma_i f_{L-2}(\vec{x}_i) \right]
$$

$$
\leq \sqrt{m} C_2 \frac{D_{L-1}^F}{2B_{L-1}^F \prod_{i=1}^{L-1} \sqrt{n_i}} \prod_{i=1}^{L} 2\sqrt{n_i} B_i^F + \sqrt{n_L} B_L^F \max_{k} \mathbb{E}_{\sigma} \left[\sup_{W_{1:L-1}} \vec{w}_{L-1,k}^{0T} \sum_{i=1}^{m} \sigma_i f_{L-2}(\vec{x}_i) \right].
$$

As in the proof for Theorem 1, we note that $\max_k \|w_{l,k}\|_2 \leq B_l^F$ and recursively apply Lemma 4 until we obtain the following bound

$$
\hat{\mathcal{R}}(\mathcal{F}_F) \leq \frac{\sqrt{2}cp C_2 \sum_{j=1}^{L} \frac{D_j^F}{2B_j^F \prod_{i=1}^{j} \sqrt{n_i}} \prod_{i=1}^{L} 2\sqrt{n_i} B_i^F}{\sqrt{m}}
$$

$$
+ \sum_{j=1}^{c} \frac{\sqrt{2}\rho \prod_{i=2}^{L} \sqrt{n_i} B_i^F}{m} \max_{k} \mathbb{E}_{\sigma} \left[\vec{w}_{1,k}^{0T} \sum_{i=1}^{m} \sigma_i \vec{x}_i \right],
$$

of which the right term is equal to zero. The result follows from Theorem 3. $\qquad\square$

C   COMPARISON WITH PREVIOUS BOUNDS

This section provides a comparison of our bounds with previous work that investigates bounding the generalisation gap,[2]

$$G = \mathop{\mathbb{E}}_{\vec{x},y}\Big[f(y, \vec{x})\Big] - \frac{1}{m}\sum_{i=1}^{m} f(y_i, \vec{x}_i).$$

In particular, we consider the main result of Bartlett et al. (2017),

$$G \leq O\left(\frac{\ln(N)}{\sqrt{m}}\rho C_2\left(\sum_{j=1}^{L}\left(\frac{D_j^{2,1}}{B_j^2}\right)^{2/3}\right)^{3/2}\prod_{j=1}^{L} B_j^2\right), \tag{6}$$

and using our notation the main result of Long and Sedghi (2019) is,

$$G \leq O\left(\sqrt{\frac{N}{m}}\rho C_2 \sum_{j=1}^{L} D_j^2 \prod_{j=1}^{L} B_j^2\right). \tag{7}$$

These bounds depend on slightly different quantities: we use N to represent the number of parameters in the network, B_j^2 and D_j^2 indicates upper bounds on the spectral norm and the the associated distance metric, and $D_j^{2,1}$ corresponds to an upper bound on distance metric induced by the matrix $(2, 1)$-norm,

$$\|W\|_{2,1} = \sum_j \sqrt{\sum_i W_{j,i}^2}.$$

In contrast to the aforementioned bounds, which were shown to be true via covering number arguments, the bounds presented in this paper are proved by directly bounding the empirical Rademacher complexity. As a consequence of these different proof techniques, the constants that appear in each bound are quite different. The covering number bound both have explicit dependences on the number of parameters in the network—with the bound of Long and Sedghi (2019) being much stronger than that of Bartlett et al. (2017). Our bound does not have this dependence, but it does incur an extra factor of two for every application of the Ledoux-Talagrand contraction inequality used in the peeling argument. This results in an explicit exponential dependence on the depth of the network. We note that recent work by Golowich et al. (2018) shows how the peeling method can be adapted to reduce this to a very mild dependence in some situations.

In addition to inspecting the constants in each bound, one can consider how the norms compare. The $(2, 1)$-norm used in Equation 6, the spectral norm bound from Long and Sedghi (2019), and the Frobenius norm used in Theorem 2 are all sensitive to the resolution of the intermediate feature maps in convolutional networks, as they do not take into account the parameter sharing in convolutional layers. In contrast, the MARS norm bound in Theorem 1 is independent of the feature map sizes. This is because the structure of W in a convolutional layer is a block matrix, where each block is, in turn, a doubly block circulant matrix, so each unique weight in the layer appears at most once in each row. The MARS norm only considers the "worst" row in the matrix, so each weight is considered at most once, irrespective of feature map resolution. As such, the MARS bound are more suited to explaining generalisation in convolutional networks. Furthermore, the way in which the norms are utilised in the bounds are different. Consider the value of each term in the summations of the bounds; for the bound of Bartlett et al. (2017), the value of each term can be greater than one, because different norms are used in the numerator and denominator. Long and Sedghi (2019) do not perform any normalisation, so each term can be greater than one. In contrast, because the numerator and denominator of the terms in the summations of our bound use the same norms, and from the definition of our hypothesis class, each term is bounded from above by one.

D   ALGORITHM PSEUDOCODE

Algorithm 1 provides high-level pseudocode demonstrating how our proposed projection functions can be integrated with a generic gradient-based optimiser.

---

[2]To simplify the comparison, we omit the terms proportional to $\sqrt{\frac{\ln(1/\delta)}{m}}$ that appear in all bounds. It should be noted that all bounds hold with probability $1 - \delta$.

Algorithm 1 A projected stochastic subgradient method for enforcing a distance constraint on each layer of a fine-tuned neural network, where f_t is the objective function evaluated on a batch of data indexed by t, update$(\cdot)$ is a weight update rule such as Adam, and π is a projection function.

   Inputs: $W_{1:L}^{(0)}$, and γ_i
   $t \leftarrow 0$
   while $W_{1:L}$ not converged **do**
      $t \leftarrow t + 1$
      $\widehat{W}_{1:L}^{(t)} \leftarrow \text{update}(W_{1:L}^{(t-1)}, \nabla_{W_{1:L}} f_t(W_{1:L}^{(t-1)}, V^{(t-1)}))$
      for $j = 1$ **to** L **do**
         $W_j^{(t)} \leftarrow \pi(W_j^{(0)}, \widehat{W}_j^{(t)}, \gamma_j)$
      end for
   end while

E   CONVERGENCE OF PENALTY METHODS

The usual motivation for using a penalty term to enforce a norm constraint is based on the equivalence of constraints and penalties in the case of linear models with convex loss functions (Oneto et al., 2016). This equivalence tells us that for each choice of penalty hyperparameter, λ, there is a corresponding setting for a constraint hyperparameter, γ, such that the penalty-based approach and the projection-based optimiser will search the same hypothesis space. Although our case is quite different from the setting of Oneto et al. (2016), since we make use of nonlinear models and apply multiple norm constraints, it is still common to apply penalties to neural networks when one wishes to encourage some sort of structure in the weights of a model. The technique used to prove the equivalence in the linear case is based on reinterpreting the penalised loss function as the Lagrangian of the constrained problem. The Karush–Kuhn–Tucker theorem states that stationary points of the Lagrangian are local optimisers of the constrained problem. The problem with applying this in a deep learning context is that convergence is almost always measured in terms of predictive performance, rather than the training loss. As a consequence, it is rare that the fine-tuned weights will be a stationary point of the Lagrangian and the constraints implied by the values chosen for the penalty hyperparameter may not be enforced. In this case, the model resulting from the training process is not guaranteed to come from the pre-specified hypothesis class. In contrast, the projection-based methods are guaranteed to provide a model from the chosen hypothesis class even if a stationary point is not found.

We explore this further by examining the when the training loss converges. Figure 4 shows plots of the training loss, as well as the training and testing accuracy, over the course of training ResNet101 and EfficientNetB0 networks on the pets dataset while applying the MARS-SP regulariser. From these plots we can see that the training and testing accuracy converges in the expected manner, but the training loss either continues to decrease throughout training or converges significantly later than the validation accuracy. This corroborates our hypothesis that users who stop based on train or validation accuracy are likely not to be at or near extrema of the loss, and hence not benefit from any enforced constraints.

F   ADDITIONAL EXPERIMENT DETAILS

We provide a summary of each datasets used in our experiments in Table 2. We selected datasets from CVonline[3] that were of a size for which fine-tuning, rather than training from scratch, is required to achieve good performance. Notably, the CUB-2011 and Stanford Dogs datasets used by prior work (Li et al., 2018; 2019) to evaluate fine-tuning regularisers are not included. This is because these two datasets overlap with ImageNet, the dataset used during pre-training. As such, we cannot be sure that the pre-trained networks have not already been trained on the test sets we are using for evaluation. Like Li et al. (2018; 2019), we also evaluate on Caltech-256. However, there are no standard training, validation, and testing splits for this dataset, and the prior work did not release their splits. As a result, we constructed our own split that consists of 30, 20, and 20 examples per class for the training, validation, and testing folds, respectively.

---

[3]http://homepages.inf.ed.ac.uk/rbf/CVonline/Imagedbase.htm

(a) ResNet-101 Loss

(b) ResNet101 Accuracy

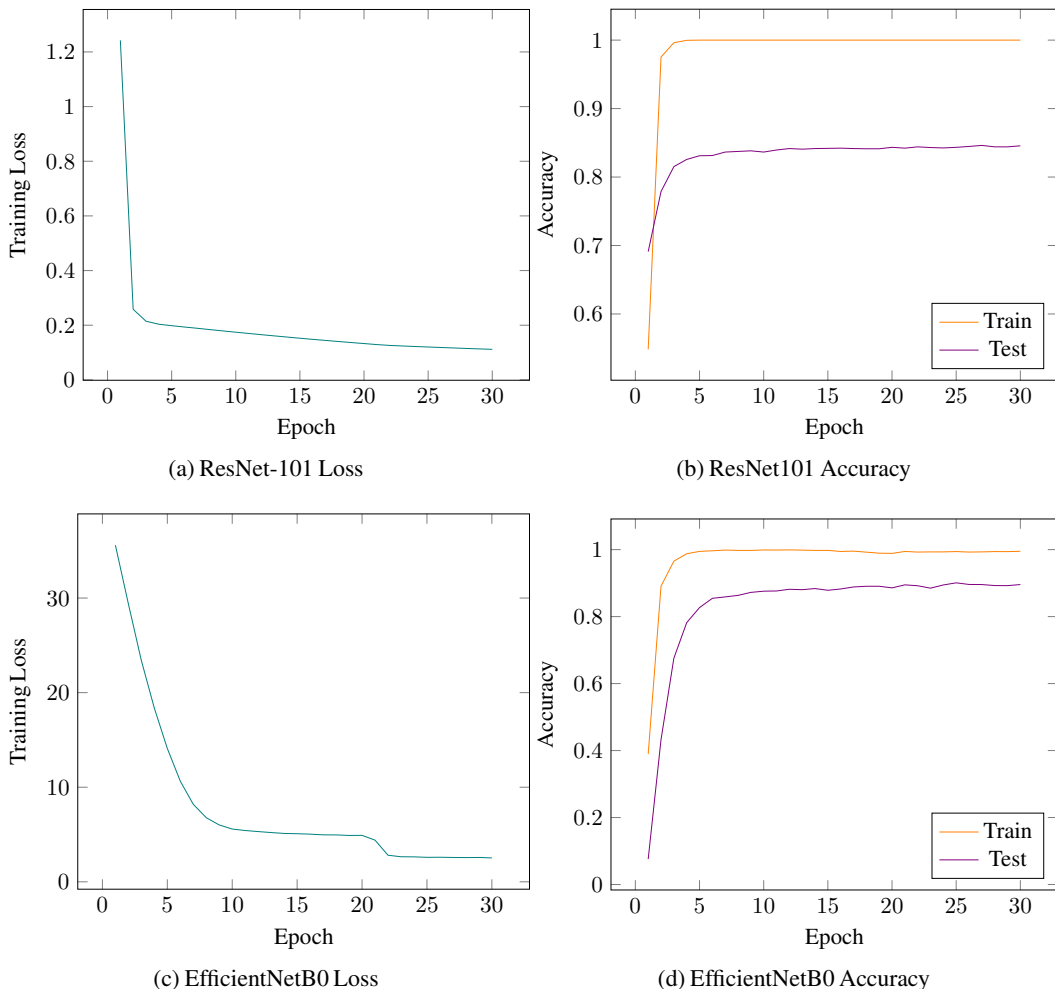(c) EfficientNetB0 Loss

(d) EfficientNetB0 Accuracy

Figure 4: Plots demonstrating the convergence properties of MARS-SP when applied to ResNet101 and EfficientNetB0 when fine-tuned on the pets dataset. Subfigures (a) and (c) demonstrate that the training loss does not necessarily converge at the same point as the accuracies in subfigures (b) and (d) have converged. Note that the learning rate is reduced by a factor of 10 at epoch 20.

All regularisation hyperparameters are tuned using the HyperOpt package developed by Bergstra et al. (2015), which uses a tree of Parzen estimators to generate promising combinations of hyperparameters based on previously observed losses. To remain comparable with ℓ^2-SP and DELTA, we make use of only two hyperparameters for each of our proposed methods: the hyperparameter corresponding to the layer trained from scratch (λ_L and γ_L for the penalty and constraint methods, respectively), and a single hyperparameter applied to all layers being fine-tuned (λ_j and γ_j). This also makes the hyperparameter optimisation process reliable, as the number of hyperparameters that must be tuned is not proportional to the number of layers being fine-tuned. HyperOpt is given 20 search iterations and hyperparameter combination with the best validation loss is selected.

F.1 Comparison using DELTA Experimental Setup

In order to facilitate a direct comparison with previously published results achieved by ℓ^2-SP and DELTA, we conduct experiments using the same experimental setup as DELTA. These experiments involve data augmentation; namely, random horizontal flips and random crops. A known issue with the experimental setup used for evaluation in prior work is the overlap between the ImageNet dataset used for pre-training and the datasets used for evaluation. We observe two interesting trends in the

Table 2: Statistics for the datasets used throughout the experiments. The Train, Validation, and Test columns contain the number of instances in each of the corresponding subsets.

Dataset	Train	Validation	Test	Classes
Aircraft (Maji et al., 2013)	3,334	3,333	3,333	100
Butterfly (Chen et al., 2018)	5,135	5,135	15,009	200
Flowers (Nilsback and Zisserman, 2008)	1,020	1,020	6,149	102
Pets (Parkhi et al., 2012)	2,000	2,000	3,390	37
PubFig (Pinto et al., 2011)	10,518	1,660	1,660	83
DTD (Cimpoi et al., 2014)	1,880	1,880	1,880	47
Caltech (Griffin et al., 2007)	7,680	5,120	5,120	256

Table 3: Comparison of fine-tuning methods on DELTA experimental setup using ResNet-101. Performance measurements for techniques other than MARS-PGM are taken from Li et al. (2019).

Method	Stanford Dogs	CUB	Food
ℓ^2-FR	84.7±0.1	61.5±0.1	64.3±0.1
ℓ^2	83.3±0.2	78.4±0.1	85.3±0.1
ℓ^2-SP (Li et al., 2018)	88.3±0.2	79.5±0.1	**86.4±0.1**
DELTA (Li et al., 2019)	**88.7±0.1**	**80.5±0.1**	86.3±0.2
MARS-PGM (Ours)	**88.7±0.2**	79.6±0.1	86.3±0.1

results presented in Table 3. First, our method is still competitive with the existing fine-tuning regularisers when using the flawed experimental protocol utilised by previous work. Second, there is a very narrow range in performance for all methods compared to the experiments conducted in Section 5. We suspect this is due to the overlap between the pre-training set and the datasets used for fine-tuning. Any strategy that keeps the weights or features sufficiently close to their pre-trained values will is likely sufficient when there is overlap between the pre-training set and the dataset used for fine-tuning.